Nexus 5x: Kernel Stack Buffer Overflow

Sagi Kedmi

IBM

Contents

1 Synopsis

Nexus 5x's kernel (msm kernel tree; android-msm-bullhead-3.10-nougat branch) exposes a sysfs file entry ("floor_vote_api") that allows a privileged attacker to trigger a kernel stack buffer overflow.

The vulnerability was verified on:

```
bullhead:/ $ getprop ro.build.fingerprint
google/bullhead/bullhead:7.0/NBD90W/3239497:user/release-keys
```

2 Stack Buffer Overflow

2.1 Vulnerable Code

All code snippets below were taken from [1].

The floor_vote_api sysfs file entry is defined as follows:

```
static DEVICE_ATTR(floor_vote_api, S_IRUGO | S_IWUSR,
                bus_floor_vote_show, bus_floor_vote_store_api);
};
```

On write() syscall, bus_floor_vote_store_api() defines a stack buffer of size 10 ("name") and in order to populate it from userspace, it uses sscanf(), insecurely.

```
static ssize_t bus_floor_vote_store_api(struct device *dev,
                         struct device_attribute *dev_attr, const char *buf,
                         size_t n)
{
    [...]
    char name[10];
    u64 vote_khz = 0;
    [...]
    if (sscanf(buf, "%s %llu", name, &vote_khz) != 2) {
        pr_err("%s:return error", __func__);
        return -EINVAL;
    }
    [...]
    return n;
}
```

2.2 Proof of Concept

In the attached `zip` archive there are both the source `poc.c` and the `aarch64` ELF binary `poc`.

The source file was compiled with:

```
$ aarch64-linux-gnu-gcc -static poc.c -o poc
```

Try the crasher on a device (you can impersonate the correct `SELinux` context and execute using it, we decided to do it with `su`):

```
$ adb push poc /data/local/tmp
$ adb shell
bullhead:/ $ su
bullhead:/ # cd /data/local/tmp
bullhead:/data/local/tmp # ./poc
```

2.3 Crash Dump

After the device crashes, `/sys/fs/pstore/console-ramoops` has the crash-dump:

```
[  774.649628] Kernel panic - not syncing: stack-protector: Kernel stack is corrupted in: ffffffc000
[  774.649628]
[  774.649672] CPU: 3 PID: 5795 Comm: p Not tainted 3.10.73-g76d746e #1
[  774.649690] Call trace:
[  774.649729] [<ffffffc000208544>] dump_backtrace+0x0/0x280
[  774.649758] [<ffffffc0002087d4>] show_stack+0x10/0x1c
[  774.649788] [<ffffffc000ed8ecc>] dump_stack+0x1c/0x28
[  774.649812] [<ffffffc000ed72e0>] panic+0x160/0x300
[  774.649842] [<ffffffc0002209a4>] __stack_chk_fail+0x14/0x18
[  774.649874] [<ffffffc000b88d74>] bus_floor_vote_store_api+0xdc/0xf0
[  774.649903] [<ffffffc0005e0598>] dev_attr_store+0x1c/0x28
[...]
```

As can be seen above - the kernel stack canary is overwritten and `__stack_chk_fail()` is callled.

File `crash` contains the entire crash-dump.

2.4 Kernel Patch

In the attached .zip there is also msm_bus_dbg_voter.patch, a patch that fixes the vulnerability.

```
diff --git a/drivers/platform/msm/msm_bus/msm_bus_dbg_voter.c b/drivers/platform/msm/msm_bus
index 2714d8a..5080e8a 100644
--- a/drivers/platform/msm/msm_bus/msm_bus_dbg_voter.c
+++ b/drivers/platform/msm/msm_bus/msm_bus_dbg_voter.c
@@ -133,11 +133,13 @@ static ssize_t bus_floor_vote_store_api(struct device *dev,
                return 0;
        }

-       if (sscanf(buf, "%s %llu", name, &vote_khz) != 2) {
+       if (sscanf(buf, "%9s %llu", name, &vote_khz) != 2) {
                pr_err("%s:return error", __func__);
                return -EINVAL;
        }

+       name[9] = '\0';
+
        pr_info("%s: name %s vote %llu\n",
                        __func__, name, vote_khz);
```

The patch was tested, i.e. kernel was compiled and flashed and the device works flawlessly.

When the poc is executed, the kernel outputs the following message to /proc/kmsg (instead of triggering a kernel stack buffer overflow):

```
[  384.555801] bus_floor_vote_store_api:return error
```

3 Attack Surface

3.1 DAC

DAC-wise, who can write to the file?

The attacker has to execute code under root within sysfs SELinux context.

```
bullhead:/sys/devices/virtual/bus-voter/bimc # ls -lZ floor_vote_api
-rw-r--r-- 1 root root u:object_r:sysfs:s0 4096 2016-01-01 08:31 floor_vote_api
```

3.2 SELinux

SELinux-wise, what contexts can write to a sysfs file ?

Looking at the aforementioned DAC, we need to find SELinux domains with allow rules that have target type sysfs with the open and write permissions on file class.

Analysing Nexus 5x's sepolicy (NRD90W) yields:

```
allow bluetooth sysfs:file { read lock getattr write ioctl open append };
allow dumpstate sysfs:file { read lock getattr write ioctl open append };
allow gpsd sysfs:file { read lock getattr write ioctl open append };
allow healthd sysfs:file { read lock getattr write ioctl open };
allow init sysfs_type:file { write lock open append relabelto };
allow netd sysfs:file { read lock getattr write ioctl open };
allow netmgrd sysfs:file { read lock getattr write ioctl open };
allow nfc sysfs:file { read lock getattr write ioctl open };
allow system_server sysfs:file { read lock getattr write ioctl open append };
allow ueventd sysfs:file { read lock getattr write ioctl open append };
allow vold sysfs:file { read lock getattr write ioctl open append };
allow wcnss_service sysfs:file { read lock getattr write ioctl open append };
```

3.3 Processes

What active processes can trigger the vulnerability?

We simply need to find which processes execute as `root` within the aforementioned `SELinux` contexts.

Analysing active processes using `ps -Z` yields:

```
u:r:init:s0       root      1     0     10128  1512  SyS_epoll_ 00004c9de4 S /init
u:r:ueventd:s0    root      310   1     5772   1172  poll_sched 00004c9e14 S /sbin/ueventd
u:r:vold:s0       root      346   1     51112  3648  hrtimer_na 7e8a3b5464 S /system/bin/vold
u:r:healthd:s0    root      382   1     6356   536   SyS_epoll_ 0000473efc S /sbin/healthd
u:r:netd:s0       root      586   1     28772  3276  binder_thr 7b5f2beb64 S /system/bin/netd
```

Code execution within any of the processes above can exploit the vulnerability.

References

[1] `android-msm-bullhead-3.10-nougat` branch. `msm_bus_dbg_voter.c`. https://android.googlesource.com/kernel/msm/+/android-msm-bullhead-3.10-nougat/drivers/platform/msm/msm_bus/msm_bus_dbg_voter.c#136. [Online; accessed 19-October-2016].